Trabalho Prático Processamento de Linguagens
Conversor Pug para HTML
Grupo 20

Bernard Georges a96326 | José Fonte a91775 | Miguel Raposo a94942

(9 de outubro de 2023)

Resumo

O presente relatório aborda o trabalho desenvolvido pelo grupo 20 no âmbito do trabalho prático da UC de Processamento de Linguagens. O tema escolhido foi "Conversor de Pug para HTML", assim sendo, o grupo desenvolveu um analisador léxico e sintático em lex e yacc, respetivamente de forma a construir um conversor funcional.

1 Introdução

O presente relatório aborda o trabalho desenvolvido pelo grupo 20 no âmbito do trabalho prático da UC de Processamento de Linguagens lecionada no curso de Licenciatura em Engenharia Informática na Universidade do Minho, no 2º semestre do 3º ano do ano letivo de 2022/23.

O tema escolhido pelo grupo 20 foi "Conversor Pug para HTML", visto tratar-se de uma linguagem utilizada na UC de Engenharia Web. De forma a entender o contexto do problema é necessário explicar os 3 conceitos em causa, o que é um conversor, o pug e o HTML.
Um conversor é um programa que permite transformar dados ou arquivos de um formato para outro. Ele realiza a conversão, ajustando a estrutura, a sintaxe ou as propriedades do ficheiro de origem para corresponder ao formato de destino desejado. Neste contexto, o programa recebe um ficheiro `.pug` e transforma-o num ficheiro `.html` .
A linguagem pug trata-se de uma linguagem de template concisa e expressiva para HTML. A linguagem fornece uma sintaxe simplificada que permite aos desenvolvedores escrever código HTML de forma mais eficiente e com menos repetição, dando, ao mesmo tempo, funcionalidades extra como lógica, includes, filtros, comentários pug, etc .

A linguagem HTML trata-se de uma linguagem de marcação utilizada para estruturar e exibir conteúdo na web. Ela define a estrutura básica de uma página da web, permitindo que os navegadores interpretem e exibam corretamente as informações na página.
Realizando este trabalho o grupo ambiciona aprofundar o seu conhecimento sobre a construção e processamento de linguagens, assim como, a utilização das bibliotecas Lex e Yacc para o auxilio no trabalho.

Desta forma, o grupo estabeleceu um conjunto de objetivos claros, definindo como Funcionalidades Básicas, tudo o necessário para o funcionamento de um programa base que transforma pug em html e como Funcionalidades Extra, recursos adicionais contidas na linguagem e descritos na documentação do pug.

Assim sendo os objetivos são :

Funcionalidades Básicas

- Abrir e ler um ficheiro `.pug`
- Análise Léxica (Lex)
  - Reconhecimento de todos os tokens - tags, literais, conteúdo, etc
  - Identificação da indentação
  - Identificação dos diferentes estados.
- Análise Sintática (Yacc)
  - Construir a Gramática da linguagem
  - Verificação de indentação - criando a hierarquia de tags correta
  - Verificação de tags fechadas
  - Associação Semântica para HTML.
- Abrir e escrever um ficheiro `.html`

Funcionalidades Extra

- Variáveis
- Atalho Doctype
- Block in a Tag
- Comentários HTML - comentários que passam para HTML
- Comentários pug - comentários que não passam
- Aplicar blocos condicionais e loops - condições *if*, ciclos *for*, ciclos *while* e ciclos *each*

2 Conceitos básicos em pug

De modo criar um conversor de pug para HTML é necessário entender alguns conceitos básicos de pug e o que o difere do HTML.

```
html(lang="en")
. head
. . title = pageTitle
. . script(type='text/javascript').
. . . if (foo) bar(1 + 5)
. body
. . h1 Pug - node template engine
. . #container.col
. . if youAreUsingPug
. . . p You are amazing
. . else
. . . p Get on it!
. . p.
. . . Pug is a terse and simple templating language with a strong focus
. . . on performance and powerful features.
```

Example 1: Ficheiro exemplo .pug

Nota: Os pontos representam a indentação do ficheiro .pug .

```
< htmllang = "en" >
< head > < title > < /title >
< script type = "text/javascript" >
if (foo) bar(1 + 5) < /script >
< /head >
< body >
< h1 > Pug − node template engine < /h1 >
< div class = "col" id = "container" >
< p > Get on it! < /p >
< p > Pug is a terse and simple templating language with a
strong focus on performance and powerful features < /p >
< /div >
< /body >
< /html >
```

Example 2: Ficheiro correspondente .html

O pug simplifica bastante o processo de escrita de HTML passando a utilizar indentação e tags simplificadas, além disso, implementa funcionalidades extra como os comentários pug/HTML, variáveis, blocos condicionais e loops, que só seriam possíveis com javascript.

2.1 Tags Simplificadas

Um dos processos mais cansativos na escrita de HTML é a abertura e fecho de todas as tags, o pug soluciona este problema utilizando uma sintaxe mais simples baseada apenas em tags de abertura. Como é possível ver pelo exemplo 1 as

tags simplificadas aliadas à indentação substituem por completo a necessidade de tags de fecho.

2.2 Indentação

A indentação é fundamental como complemento das tags simplificadas definindo a estrutura hierárquica do código e eliminando a necessidade de uma tag de fecho.

2.3 Funcionalidades Extra

O pug permite implementar algumas funcionalidades extra como lógica, blocos de texto, comentários, etc. dessa forma, é necessário consultar a documentação do pug para entender a sintaxe específica de cada funcionalidade.

3 Proposta de solução

Para solucionar o problema em questão o grupo propõe um programa que se materializa em dois ficheiros - um analisador léxico e outro um analisador sintático.

O objetivo da analisador léxico é dividir o código fonte em unidades léxicas chamadas de "tokens". Esses tokens representam os elementos básicos da linguagem, como palavras-chave, identificadores, operadores e símbolos. O analisador léxico lê o código Pug carácter-a-carácter e identifica os tokens correspondentes, removendo espaços em branco. A saída desse processo é uma sequência de tokens, que servirá de entrada para a próxima etapa.

O objetivo do analisador sintático é verificar se a sequência de tokens gerada pela análise léxica está correta e segue as regras gramaticais da linguagem. O analisador sintático utiliza uma gramática formal para realizar essa verificação e construir uma árvore de análise sintática, conhecida como árvore de derivação. Essa árvore estrutura e representa a hierarquia dos elementos no código Pug.

3.1 Analisador Léxico (Lex)

Objetivo : Criar uma stack de *Tokens*

No analisador léxico, o grupo deve explicitar o regex de cada *token* possível no ficheiro input e retornar *stack* com todos os *tokens* identificados.

Dadas as funcionalidades extra, em certos tokens o analisador léxico deve analisar as indentações, os comentários e outros casos específicos.

3.2 Analisador Sintático(Yacc)

Objetivo : Construção da Árvore de código HTML

No analisador sintático, o grupo deve construir a gramática da linguagem com os tokens enviados pelo lexer de forma a retornar um texto funcional de HTML, para isso o analisador deve ter em conta as tags previamente abertas, analisando a cada passo o seu estado corrente.

Dadas as funcionalidades extra, é adicionada mais complexidade ao analisador sintático. Por exemplo, no caso dos blocos condicionais e loops o analisador deve processar o código, isto é, executar a lógica escrita em pug, outro exemplo é o caso das variáveis, mantém-nas num dicionário.

4 Implementação

Todo o código com a solução implementada encontra-se num ficheiro anexado ao relatório, no entanto, pode-se realçar os resultados de cada parte do código.

4.1 Analisador Léxico (Lex)

```
LexToken(TAG,('html', 2),1,3)
LexToken(AP,'(',1,7)
LexToken(STRING,'lang="en"',1,8)
LexToken(FP,')',1,17)
LexToken(TAG,('head', 4),1,23)
LexToken(TAG,('title', 6),1,34)
LexToken(EQ,'pageTitle',1,39)
LexToken(TAG,('script', 6),1,57)
LexToken(AP,'(',1,63)
LexToken(STRING,"type='text/javascript'",1,64)
LexToken(FP,')',1,86)
LexToken(STRING,'if (foo) bar(1 + 5)',1,89)
LexToken(STRING,'body',1,119)
LexToken(TAG,('h1', 6),1,134)
LexToken(STRING,'Pug - node template engine',1,137)
LexToken(HT,('div', 6),1,170)
LexToken(ID,'id=container',1,171)
LexToken(CLASS,'class="col"',1,180)
LexToken(IF,('if', 10),1,195)
LexToken(VALUES,'youAreUsingPug',1,198)
LexToken(TAG,('p', 14),1,227)
LexToken(STRING,'You are amazing',1,229)
LexToken(ELSE,('else', 10),1,255)
LexToken(TAG,('p', 14),1,274)
LexToken(STRING,'Get on it!',1,276)
LexToken(END,10,1,297)
LexToken(TAG,('p', 10),1,297)
LexToken(STRING,'Pug is a terse and simple templating language with a',1,300)
LexToken(STRING,'strong focus on performance and powerful features',1,368)
LexToken(END,'',1,431)
```

Figura 1: Stack de tokens do exemplo 1

4.2 Analisador Sintático (Yacc)

```
<html lang="en">
<head>
<title>
Hello World</title>
<script type='text/javascript'>
if (foo) bar(1 + 5)</script>
</head>
<body>
<h1>
Pug - node template engine</h1>
<div id=container class="col">
<p>
You are amazing</p>
<p>
Pug is a terse and simple templating language with a strong focus on performance and powerful features</p>
</div>
</body>
</html>
```

Figura 2: Código HTML resultante

5 Testes e Qualidade do Software

De modo a provar a qualidade do software, o grupo executou vários testes com diferentes objetivos:

- teste1.pug : testa as funcionalidades básicas (exemplo do professor) - e algumas extra pois implementa variáveis e block in tag
- teste2.pug : testa comentários HTML/PUG e Atalho Doctype.
- teste3.pug : testa todas as funcionalidades extra - adiciona os blocos condicionais e loops.

Executados os diferentes testes, todos indicam resultados positivos e que cumprem os requisitos para um excerto funcional de HTML.

6 Conclusão

Em resumo, este trabalho de desenvolvimento de um conversor Pug para HTML, utilizando Lex e Yacc, proporcionou uma solução eficiente para transformar templates Pug em código HTML válido. A combinação destas ferramentas permitiu uma análise detalhada do código fonte, garantindo a correta estruturação em tokens e verificação a gramática. Com este conversor, conseguimos produzir código HTML bem-estruturado e semanticamente correto, contribuindo para o desenvolvimento eficiente de projetos web.

Como trabalho futuro, o grupo considera que tudo pode ser melhorado e este trabalho não é exceção, assim sendo, teremos a possibilidade de implementar mais funcionalidades descritas na documentação do pug e corrigir eventuais erros.